This document is published in: Ana Iglesias, Paloma Martínez, Ricardo Aler, Fernando Fernández, (2009). Learning teaching strategies in an Adaptive and Intelligent Educational System through Reinforcement Learning. In *Applied Intelligence,* August 2009, Volume 31, Issue 1, pp 89-106. DOI: http://dx.doi.org/10.1007/s10489-008-0115-1

# Learning teaching strategies in an Adaptive and Intelligent Educational System through Reinforcement Learning

**Ana Iglesias · Paloma Martínez · Ricardo Aler · Fernando Fernández**

**Abstract** One of the most important issues in Adaptive and Intelligent Educational Systems (AIES) is to define effective pedagogical policies for tutoring students according to their needs. This paper proposes to use Reinforcement Learning (RL) in the pedagogical module of an educational system so that the system learns automatically which is the best pedagogical policy for teaching students. One of the main characteristics of this approach is its ability to improve the pedagogical policy based only on acquired experience with other students with similar learning characteristics. In this paper we study the learning performance of the educational system through three important issues. Firstly, the learning convergence towards accurate pedagogical policies. Secondly, the role of exploration/exploitation strategies in the application of RL to AIES. Finally, a method for reducing the training phase of the AIES.

**Keywords** Intelligent tutoring systems · Adaptive and Intelligent Educational Systems · Applied artificial intelligence · Reinforcement Learning · Learning pedagogical strategies

A. Iglesias (✉) · P. Martínez · R. Aler · F. Fernández
Computer Science Department, Universidad Carlos III de Madrid, Avda. de la Universidad, 30, 28911 Leganés, Madrid, Spain
e-mail: aiglesia@inf.uc3m.es

P. Martínez
e-mail: pmf@inf.uc3m.es

R. Aler
e-mail: aler@inf.uc3m.es

F. Fernández
e-mail: ffernand@inf.uc3m.es

## 1 Introduction

Distance education is currently a significant research and development area. In recent years, distance educational systems have been improved, opening new perspectives on different ways to teach. The use of the Internet as a tool for educational systems helps to avoid physical barriers, classrooms access and incompatibilities resulting from different student system platforms.

Traditionally, the courses in Web-based systems consist of static hypertext pages with no student adaptability. However, since the 1990s, researchers began to incorporate adaptability into their systems [1]. Web-based Adaptive and Intelligent Educational Systems (Web-based AIES) are distance educational systems based on the Internet.

One of the main problems in AIES is to determine how to adapt the curriculum sequence to each student according to their learning characteristics. Several Machine Learning (ML) techniques are used in AIES in order to choose the best pedagogical strategy to be applied in each moment, like neural networks [2], Bayesian networks [3], etc. In a previous paper [4], we propose to use a knowledge representation based on Reinforcement Learning (RL) [5] that allows AIES to adapt tutoring to students' needs. In this way, the system could be able to improve its pedagogical policy, sequencing the system's content in an accurate way according the current student's needs based only on the student's performance, lesson objectives and the relationships among course modules. This proposal avoids the definition of every static and predefined pedagogical policy for each student. We call the system implemented according this proposal RLATES (Reinforcement Learning in Adaptive and intelligent Educational Systems).

However, this feature could be a problem in Web-based educational systems, because adaptation based only on pre-

vious interactions with other students implies that when the system is in its learning phase, it is not teaching students in the best way. So the system development requires two phases: the *Training* phase, where the system explores new alternatives in order to teach the system contents; and the *Use* phase, where the system uses the experience previously acquired from interactions with other students with similar learning characteristics.

That is why in this paper we deeply analyze the viability of RLATES under a simulated and theoretical point of view before using RLATES for real situations. This paper presents the study of the learning performance of RLATES through three important issues. Firstly, we demonstrate that the system is able to converge into a good pedagogical strategy when it interacts with simulated students with different learning characteristics. The quality of the obtained policy is measured in: (i) the number of actions that the system needs to carry out to teach all of the contents to the student; (ii) the number of students required in the training phase. Secondly, two typical exploration/exploitation strategies in the reinforcement learning bibliography have been studied in order to decide which one is more appropriate to be used in Adaptive and Intelligent Educational Systems. Finally, we have studied how to reduce the *Training* phase of the system by initializing the system with pedagogical information coming from interactions with other simulated students. This third study is crucial for the viability of the system in real situations, because it could reduce the number of students that the AIES is not teaching in the best way.

The experiments show that the system learns to teach by trial and error at the same time as simulated students learn the AIES's material. For the experiments, an example of a Database Design (DBD) AIES is presented.

The paper is organized as follows: firstly, Sect. 2 describes the main characteristics of Adaptive and Intelligent Educational Systems and Reinforcement Leaning. Then, Sect. 3 describes the architecture of the DataBase Design Adaptive and Intelligent Educational System (DBD AIES) used in this work. Section 4 defines the DBD AIES system as a RL problem. Then, Sect. 5 summarizes experimental results. Finally, the main conclusions and further research of this work are given.

## 2 Background

The main characteristics of Web-based Adaptive and Intelligent Educational Systems and the Reinforcement Learning Model are described in this section.

### 2.1 Web-based Adaptive and Intelligent Educational Systems

A Web-based AIES adapts the policy to teach to each student according to their learning characteristics by applying artificial intelligence techniques. These systems have three useful benefits: classroom independence, platform independence, and intelligent adaptability to the learners according to their needs.

The Web-based AIES systems are not totally a new type of systems, but they stem from *Intelligent Tutoring Systems (ITS)* and *Adaptive Hypermedia Systems (AHS)*. Intelligent Tutoring Systems (ITSs) are computer-aided instructional systems with models of instructional content that specify *what* to teach and teaching strategies that specify *how* to teach [6]. Adaptive Hypermedia Systems adapt the content of a hypermedia page to the student's goals, knowledge, preferences and other student information for each individual student interacting with the system [1].

Both ITSs and AHS share one of the oldest problems in educational systems: what to teach next and how to do it. In Intelligent Tutoring Systems, *Curriculum Sequencing* has been widely studied and specifies the individually planned sequence of knowledge units and learning tasks (examples, questions, problems, etc.) that are most suitable according to the student learning needs at each moment. On the other hand, the Adaptive Hypermedia Systems study the *Adaptive Navigation Support* technology *(ANS)*. This technology provides the student with hyperspace orientation and navigation by changing the appearance of the pages and their visible links.

In this paper, we address this problem again by applying Reinforcement Learning so that the system could learn accurate pedagogical strategies to teach students according to their learning characteristics.

#### *2.1.1 Pedagogical strategies*

A pedagogical strategy specifies how to sequence, how to provide feedback to students and how to show, explain or summarize the system content [7]. Most of the current distance educational systems define their pedagogical strategies as predefined action plans showing the system material to each student according to their learning characteristics [8]. The heterogeneity of the students interacting with the system is one of the main problems in adaptive educational systems, because the system has to adapt to each student at each interaction step.

To choose the appropriate pedagogical strategy at each moment is not an easy task. It is necessary to define the teaching strategies according to each student at every step of the interaction and it is also necessary to specify when and where the strategies are different [9]. Moreover, if a pedagogical strategy has been chosen and it is not the most appropriate strategy to be applied at this moment, we have to discover why, and how to solve it.

How to choose the best pedagogical strategy to be applied at each moment has been widely studied, but it usually

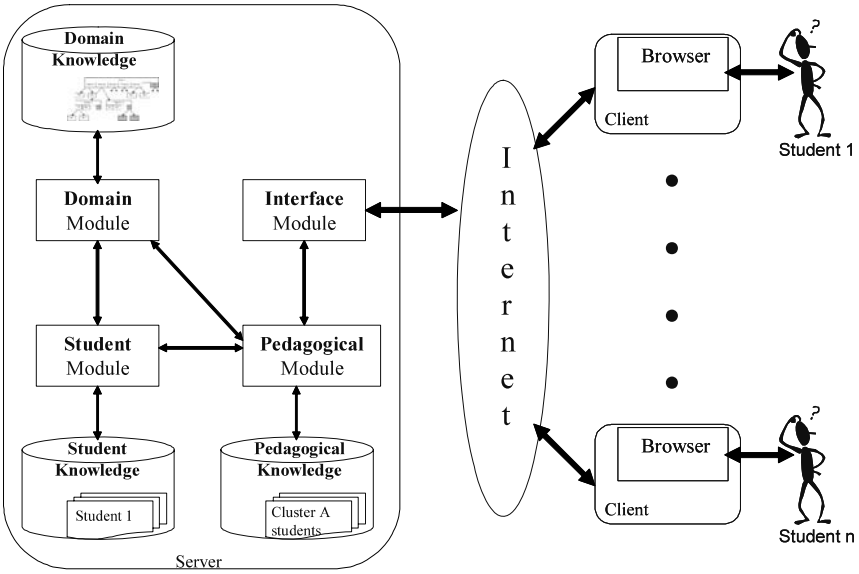


**Fig. 1** AIES architecture

has a high cost. In the first educational systems, the students were responsible for their own learning, such as in PMS system [10]. Nowadays, artificial intelligence techniques are applied in order to solve the problems derived from choosing the pedagogical strategy; for instance, semantic nets have been applied in the MENO-TUTOR system [11], neural nets have been used in the UNIMEM system [2], bayesian networks have been applied in DT Tutor [3] and reinforcement learning model have been used to define the user model in the ADVISOR system [12].

#### 2.1.2 *Architecture of Adaptive and Intelligent Educational Systems*

A typical structure of an ITS, and hence, of an AIES, is made up of four well differentiated modules (see Fig. 1) [13]:

- The *student module* manages all important information about the student in the learning process: student knowledge, personal characteristics, historical behaviour, learning aptitudes, etc.
- The *interface module* facilitates the communication between the AIES and the student. Therefore, it has to have high usability, to be intuitive and not ambiguous, etc.
- The *domain module* contains all characteristics of the knowledge to teach. It is sometimes called the *expert module* because some systems use it like an *expert system* [14], storing information on the topics, tasks, relationships between them, difficulty of each task, etc.
- The *pedagogical module* decides *what*, *how* and *when* to teach the domain module contents, making better pedagogical decisions according to the student needs. Some researchers call it the *tutor module*, because it compares the student knowledge with the domain knowledge and then chooses the pedagogical strategies to teach the students.

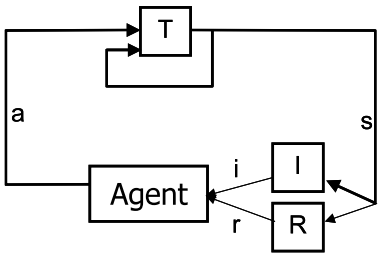


**Fig. 2** Reinforcement learning architecture

### 2.2 Reinforcement Learning

Reinforcement learning deals with agents connected to their environment via perception and action. At each step of the interaction, the agent observes the current state, $s$, and chooses an action to be executed, $a$. This execution produces a state transition and the environment provides a reinforcement signal, $r$, indicating the quality of the state transition when the agent solves the task. The final goal of the agent is to behave choosing the actions that tend to increase the long-run sum of values of the reinforcement signal, $r$, learning its behavior by systematic trial and error, guided by a wide variety of algorithms [15].

RL problems are usually described by *Markov Decision Processes* (MDP) [16]. The components of an MDP are described in Fig. 2. The $I$ function indicates how the agent perceives the current state of the environment. The dynamic of the domain is defined by using the state transaction function, $T$, and the reinforcement signal, $R$:

- $R : S \times A \rightarrow R$, where a reinforcement signal is provided for each state and each action and is defined in (1),

$$R(s,a) = R^{a}_{ss'} = E\{r_{t+1} \mid s_t = s, a_t = a\} \tag{1}$$

- $T : S \times A \times S \rightarrow P$, where for each state and each action, a probability distribution is provided. The function is shown in (2), where $T(s, a, s')$ indicates the probability of transiting to state $s'$ when the agent was in state $s$ and the action $a$ is carried out.

$$T(s, a, s') = P^{a}_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\} \tag{2}$$

Then, the main goal of the agent is to find a policy, $\pi$, that tends to maximize the long-run sum of values of the reinforcement signal.

One of the most important properties of the MDP is that the next state depends only on the current state and action and is independent of the preceding states and actions. In (3) this property (called the *Markov* property) is captured.

$$\begin{aligned}&\Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\} \\ &\quad = \Pr\{s_{t+1} = s', s_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots, \\ &\qquad s_0, a_0, r_0\}\end{aligned} \tag{3}$$

### 2.2.1 Utility functions

Most of the reinforcement learning algorithms are based on the utility functions, which estimate how good the action policy is. The action-value function described in (4), for instance, is defined as the expected reward that will be received if the agent is located in a state *s,* the first action that it carries out is action $a$, and, then, it follows the action policy $\pi$.

$$\begin{aligned}Q^{\pi}(s, a) &= E_{\pi}\{R_t \mid s_t = s, a_t = a\} \\ &= E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a\right\}\end{aligned} \tag{4}$$

This function introduces the discount parameter, $\gamma$, to differentiate when the reinforcements have been obtained.

Therefore, the goal of the learning process is to find the policy such that it maximizes this function. In other words, to obtain the optimal value-action function, denoted by $Q^*(s, a)$, such that $Q^*(s, a) \geq Q(s, a)\ \forall a \in A, s \in S$.

From the optimal $Q$-function, the optimal policy can be obtained as is shown in (5).

$$\pi^{+}(s) = \arg\max_{a} Q^*(s, a) \tag{5}$$

### 2.2.2 Q-learning: an algorithm for learning $Q^*$

The $Q$-learning algorithm [17] is based on the *value-action* function, $Q(s, a)$, to compute its optimal action policy. The definition of this function for non-determinist environments is described in (6), where the $\gamma$ parameter controls the relative importance of future action rewards with respect to new ones, and $\alpha$ parameter is the learning rate, which indicates how quickly the system learns. In Table 1, the *Q-learning* algorithm is described.

**Table 1** $Q$-learning algorithm

| $Q$-learning algorithm |
|---|
| ■ For each pair ($s \in$ S, $a \in A$), initialize the table entry $Q(s, a) = 0$. |
| ➢ Observe the current state, $s$ |
| ➢ Do forever |
| – Select an action, $a$, following an exploitation/exploration strategy, and execute it |
| – Receive the immediate reward, $r$ |
| – Observe the new state, $s'$ |
| – Update the table entry for $Q(s, a)$ as follows: $Q(s, a) = (1 - \alpha)Q(s, a) + \alpha\{r + \gamma \max_{a'} Q(s', a')\}$ |
| – Set $s$ to $s'$ |

$$Q(s, a) = (1 - \alpha)Q(s, a) + \alpha\left\{r + \gamma \max_{a'} Q(s, a'\right\} \tag{6}$$

### 2.2.3 Exploration versus exploitation

Choosing the next action to carry out is very important for the convergence of the *Q-learning* algorithm. The quality of the selected exploration/exploitation strategy is determinant in the efficiency and efficacy of the learning process. The exploitation strategy tries to use acquired previously knowledge, and the exploration strategy tries to find new alternatives.

A great variety of exploitation strategies could be used in reinforcement learning problems [18]. For instance, the $e$-greedy strategy tries to select the action with the greater $Q$ function value, with a probability of $e$. It chooses the action randomly with probability $(1 - e)$.

Another very common exploitation strategy is the Boltzmann strategy, that estimates the probability of choosing the action $a$ according to the (7), where $\tau$ is a positive parameter called the *temperature.* If the temperature is high, all the probabilities of the actions have similar values and if the temperature is low, it causes a great difference in the probability of selecting each action.

$$P(a) = \frac{e^{\frac{Q_t(a)}{\tau}}}{\sum_{b=1}^{n} e^{\frac{Q_t(b)}{\tau}}} \tag{7}$$

In Sect. 4 we will demonstrate the influence of these strategies when the AIES is learning a pedagogical policy.

## 2.3 Related work

How to apply artificial intelligence techniques to automatically build or maintain the different modules of intelligent

tutoring systems has been widely studied. The objective is to emulate the human behaviour while teaching only one student in a classroom.

All the modules of an ITS has been studied under the perspective of Artificial Intelligence. The domain module is often described as part of an expert system, using *frames* in TEX-Sys tutor [33] or *bayesian networks* [34] between others.

The interface module can use artificial intelligence techniques in order to attract the student attention. For instance, *natural language* techniques has been applied in dialogue systems as PACT Geometry Tutor [35] or CycleTalk system [36], and *annotation* or *search* techniques has been applied in ELM-ART system does [37], etc.

Different techniques have been used in the student module, as *stereotypes* in [38], the *overlay* model in [39] or *bayesian networks* in [34]. *Inductive learning* has been used in ACM [40], ASSERT [41] or MEDD [42], with the objective of learning some properties of the students. *Deductive and supervised learning* has been applied in PIXIE [43] and the Hoppe's system [44], for instance.

Finally, the pedagogical module has been defined as an automated *planning* problem in WIP/PPP [45]. It also has been implemented with a set of *production rules in* LISP tutor [46], or with *constraint rule*, *s* as KERMIT system does [47], combining *production rules* with *neural networks*. However all of then need to predefine a huge amount of information. WIP/PPP system, for instance, needs to predefine presentation strategies (how to select relevant content, how to structure selected content and which medium to use for conveying a content). On the other hand, the systems using production rules had to manually predefine the pedagogical strategies. To define manually this information is very expensive, because it is necessary to define a huge amount of pedagogical strategies or presentation strategies in order to adapt to each student in each moment of the interaction and sometimes is difficult to add expert knowledge in the system. As we have showing in the paper, Reinforcement Learning can help to define these strategies automatically.

Other advantage of applying reinforcement learning in the pedagogical module is that this model absorbs noise. For instance, it is not easy to detect current student knowledge about a specific topic; if the system is wrong about this, the reinforcement learning model could leave out this error.

The reinforcement learning model has been previously applied to the student module in ADVISOR system [48], using simulated students to evaluate the approach, To apply reinforcement learning in this module has a main problem: the student has a lot of learning characteristics to take into account. This is a problem in the reinforcement learning paradigm because the number of states of the system increases, making more difficult the system to learn. Moreover, Beck's system learns very slow due to the system is learning from the interaction with other students with similar learning characteristics, being difficult to find enough students to learn appropriately. In this work we propose to apply the reinforcement learning model in the pedagogical module, taking only into account the student knowledge about each topic in the domain. In this way, the system is able to adapt to each student interacting dynamically.

On the other hand, in Beck's system the content is sequenced in coarse grain (knowledge topics), pointing out that it is more effective that sequencing in fine grain (exercises, problems, etc.). In this paper we show the advantages of sequencing the domain content structured in fine grain, taking into account tasks sequencing (introductions, examples, problems, etc.), not only topics sequencing, choosing at the same time the presentation format of the content.

Finally, ADVISOR uses the *e-greedy* exploration-exploitation learning policy, receiving an immediate reinforcement. A complete study of the viability of RLATES has been carried out in this paper, analysing different exploration-exploitation techniques and different learning variables trying to adapt to each student as soon as possible. The application of *Boltzmann* exploration-exploitation strategy could be an advantage in tutoring systems, providing the students information about the probability of choosing the next content of the system. Moreover, a study of reducing the system training phase has been carried out.

## 3 Architecture of the proposed Adaptive and Intelligent Educational System

The system presented in this paper (RLATES: Reinforcement Learning in an Adaptive and Intelligent Educational System) adopts the typical architecture of the AIES described in Sect. 2.1.2, where four modules are used: the *domain module*, the *student module*, the *interface module* and the *pedagogical module*.

AIES knowledge is stored in the *domain module*. Figure 3 shows a proposal of a hierarchical knowledge structure for a Database Design domain, consisting of knowledge on the Extended Entity/Relationship Model [19]. In the knowledge tree, each topic is divided into subtopics, and these into other subtopics, and so on. At the same time, each node of the tree contains sets of definitions, examples, problems, exercises, etc. in several formats (image, text, video, etc.).

On the other hand, it is necessary, for the effectiveness of this proposal, to construct a good *student module* where the current level of knowledge of the student in each moment of the interaction with the AIES is the main characteristic

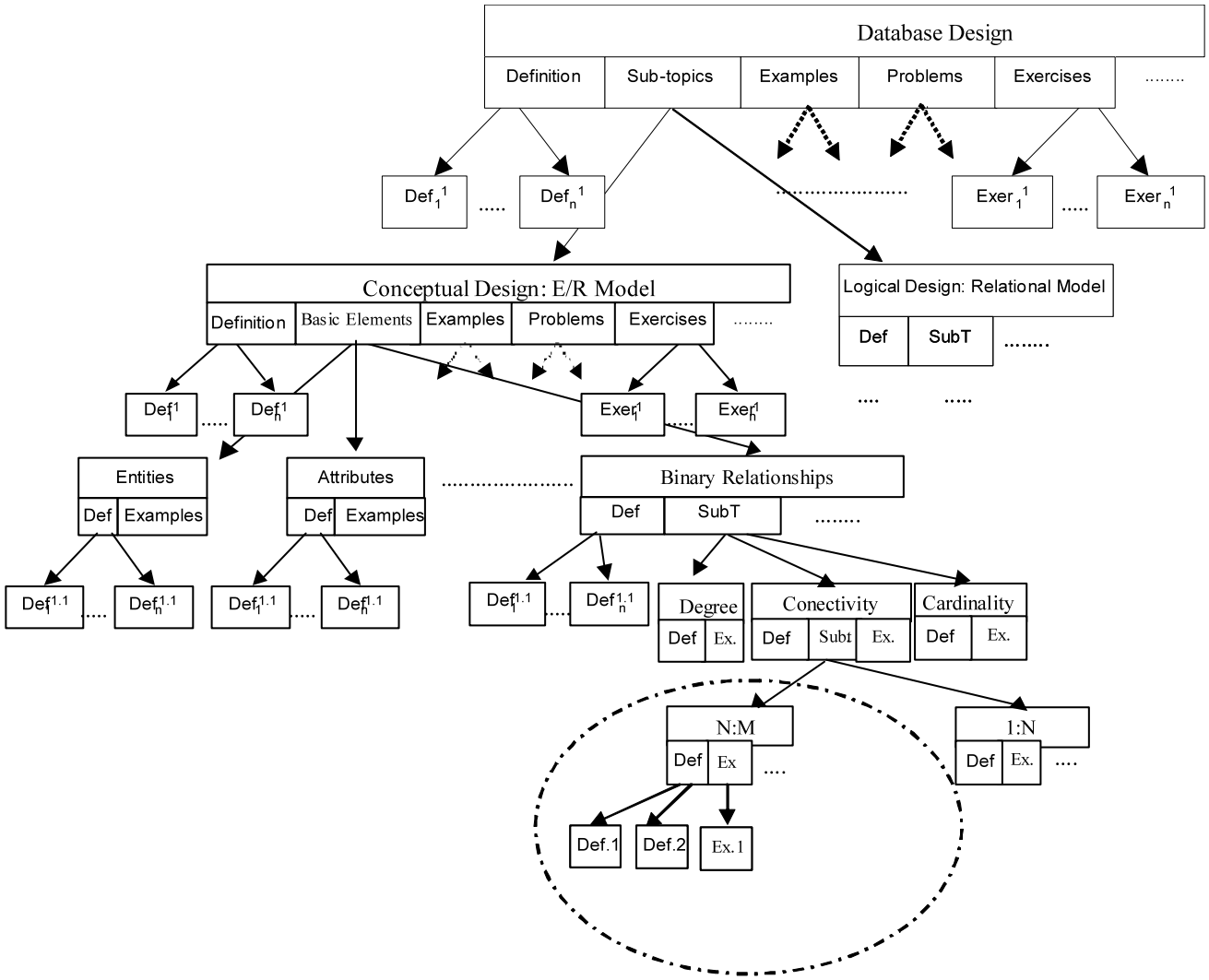


**Fig. 3** Example of Database Design AIES knowledge model

to model. Any classic student model technique that could represent this information could be used for the AIES. For instance, we can use *Stereotypes* [20], the *Overlay* model [21] or *Bayesian Networks* [22] in order to represent this information. The *Overlay* model has been used to implement the student model in RLATES.

The interface module of the AIES adapts the content of a hypermedia page to each student's goals, knowledge, preferences and other student information, changing the appearance of the pages and their visible links.

Finally, in the *pedagogical module* the AIES finds the best way to teach the *knowledge items,* corresponding to the internal nodes of the tree (topics), to the current student. The definition of this problem as a Reinforcement Learning problem is addressed in Sect. 4, where a guide to apply Reinforcement Learning model in Adaptive and Intelligent Educational Systems is provided.

## 4 Application of Reinforcement Learning in AIES

In this section, we briefly define how Reinforcement Learning is applied in Adaptive and Intelligent Educational Systems and how an RL algorithm (the $Q$-learning algorithm) is adapted to the tutoring domain. More information on this issue can be found in [4].

### 4.1 Reinforcement learning components

Next, the components of a reinforcement learning problem in the Database Design AIES environment are briefly defined:

(1) *Set of states (S):* A state is defined as the description of the current student knowledge. It is represented by a vector which stores as many representative values of learner knowledge items (internal node of the knowledge tree) as the student should learn. In order to simplify the example,

**Table 2** Adaptation of the $Q$-learning algorithm to AIES domain

| $Q$-learning algorithm | $Q$-learning adapted to AIES domain |
|---|---|
| ■ For each pair ($s \in S$, $a \in A$), initialize the table entry $Q(s, a)$. | ■ For each pair ($s \in S$, $a \in A$), initialize the table entry $Q(s, a)$. |
| ■ Observe the current state, $s$ | ■ Test the current student knowledge, obtaining $s$ |
| ■ Do forever | ■ While $s$ was not a goal state do: |
| ◆ Select an action, $a$, following an exploitation strategy, and execute it | ◆ Select a knowledge tree leave, $a$, to show to the student, based in an exploitation strategy derived from $Q$. |
| ◆ Receive the immediate reward, $r$ | ◆ Receive the immediate reward, $r$. A positive reward is received when the AIES goal is achieved. A null reward is obtained in any other case. |
| ◆ Observe the new state, $s'$ | ◆ Test the current student knowledge, $s'$ |
| ◆ Update the table entry for $Q(s, a)$ as follows: | ◆ Update the table entry for $Q(s, a)$, that estimates the usefulness of executing the a action when the student is in a particular knowledge state: |
| ◆ $Q(s, a) = (1 - \alpha)Q(s, a) + \alpha\{r + \gamma \max'_a Q(s', a')\}$ | ◆ $Q(s, a) = (1 - \alpha)Q(s, a) + \alpha\{r + \gamma \max_{a'} Q(s', a')\}$ |
| ◆ Set $s$ to $s'$ | ◆ Let us $s$ the current student knowledge state, $s$'. |

it is supposed that these values are defined in the set {0, 1}. *Zero* indicates that the student does not know the item, and *one* indicates that the item has been correctly learned. The items of the knowledge tree are enumerated in a *pre-ordered* way.

(2) *Set of actions (A):* The actions that the tutor can carry out are those that teach the AIES subject, in other words, to show the knowledge items defined in the leaves of the knowledge tree.

(3) *Perception of the environment* ($I : S \rightarrow S$)*:* This function indicates how the AIES perceives the state the student is in. An AIES could perceive the knowledge state of the student through evaluating his/her knowledge by tests or exams.

(4) *Reinforcement* ($R : S \times A \rightarrow R$)*:* This function defines the reinforcement signals (rewards) provided by the environment. This reinforcement function supplies a maximum value to meet the goals of the tutor; that is to say, when the student learns the totality of the contents of the AIES.

(5) *Value-action function* ($Q : S \times A \rightarrow R$)*:* This function estimates the usefulness of showing leaves of the tree to a student when s/he is in a certain knowledge state. This function provides an evaluation method for the tutor action policies. Therefore, the goal of the learning process is to find the policy such that it maximizes this function, in other words, to obtain the optimal value-action function.

In Table 2, an explanation is given regarding how the $Q$-learning algorithm is adapted to the tutor domain. This is based on the definition of the Reinforcement Learning components for an Adaptive and Intelligent Educational System. An example of the learning to teach process of a Database Design AIES with the *Q-learning* algorithm appears in [23].

### 4.2 System functional phases

For the correct working of the system, three functional phases have been defined, related to the exploration/exploitation strategies (see Sect. 2.2.3): student clustering, system training and system use. These phases have been represented in Fig. 4.

1. *Student Clustering.* Although the system is able to adapt its curriculum sequencing depending on each student's learning characteristics by updating its value-action function, $Q(s, a)$, it is recommended that the students are clustered according to their learning characteristics before training the system. In this way, the system will keep a $Q$ table for each student cluster, improving the system adaptation to each group of students. Moreover, the system will converge faster and better, learning better pedagogical strategies. Notice that this phase is not necessary, but it is recommended in order to improve the system adaptation. In this paper, we prove that the more

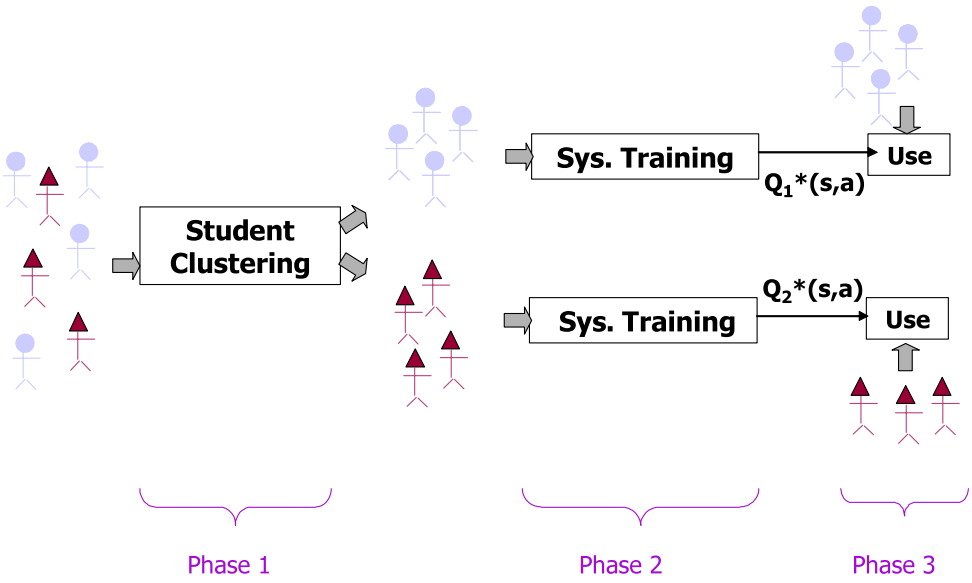


**Fig. 4** System functional phases

homogeneous the cluster, the faster and better the system learning convergence is. A great variety of student models and techniques has been studied [24], and any classification technique, like C4.5 [25] or Reinforcement Learning techniques [12], could be used to cluster students according to their learning characteristics.

2. *System training.* In this phase, the system explores new pedagogical alternatives in order to achieve the goal, showing the course content to the students in different sequences. At the same time, the system interacts with the students and updates its $Q$ table. In this way, the system learns a good teaching strategy based only on previous interactions with other students with similar learning characteristics (students in the same cluster).
3. *System use.* Finally, once the system has converged to a good pedagogical strategy, this strategy can be used to teach other students with similar learning characteristics.

Although the students' interaction with the system has been divided into two phases (training and use phases), the system never ends its learning process (it is always updating its $Q$ table). In this paper, this division enhances the difference between exploration (at the training phase) and exploitation (at the use phase). The ideal situation for the system is when the training phase is reduced as much as possible. This paper studies how to minimize the training phase in Sect. 5.

## 5 Experiments

Numerous experiments are necessary in order to obtain general conclusions on evaluating an educational system, and therefore, a lot of students are needed to interact with the system. Normally, it is difficult to convince one student to interact with a system in evaluation. We are unlikely to convince as many as we need to complete the experiments. Thus, most of the pedagogical systems use simulated students in its evaluation, such as the ADVISOR system [12], the STEP system [26] and the VanLehn system in [27].

### 5.1 Experimentation setup

In this section we present the experiments carried out with simulated students on the AIES learning process in order to test five issues. (1) the system converges to a policy; (2) the policy obtained only requires a few actions to teach the AIES content to the current student, that it is to say, the system converges to a near optimal policy; (3) the AIES only needs a few students to converge to get close to the optimal pedagogical policy; (4) important differences at the system convergence of the system are found when different exploration/exploitation strategies are chosen; (5) the *training* phase of the system can be minimized if the $Q$ table is initialized. We have divided the experimentation into three different sequences. Firstly, in Sect. 5.2 the system learning convergence is studied and the first three issues are validated using different learning parameters. Next, in Sect. 5.3, issue (4) is tested studying the role of exploration/exploitation strategies in the application of RL to AIES. And finally, in Sect. 5.4, issue (5) has been studied, studying a method for reducing the *Training*.

The domain of an educational system is stochastic, because the students could pass or not pass a test depending on several characteristics. Because of this, the experimentation with simulated students has been carried numerous times and statistical measures like average and standard deviation are used in order to analyze the results [28]. Moreover, series of $t$-tests in every experiment presented in this

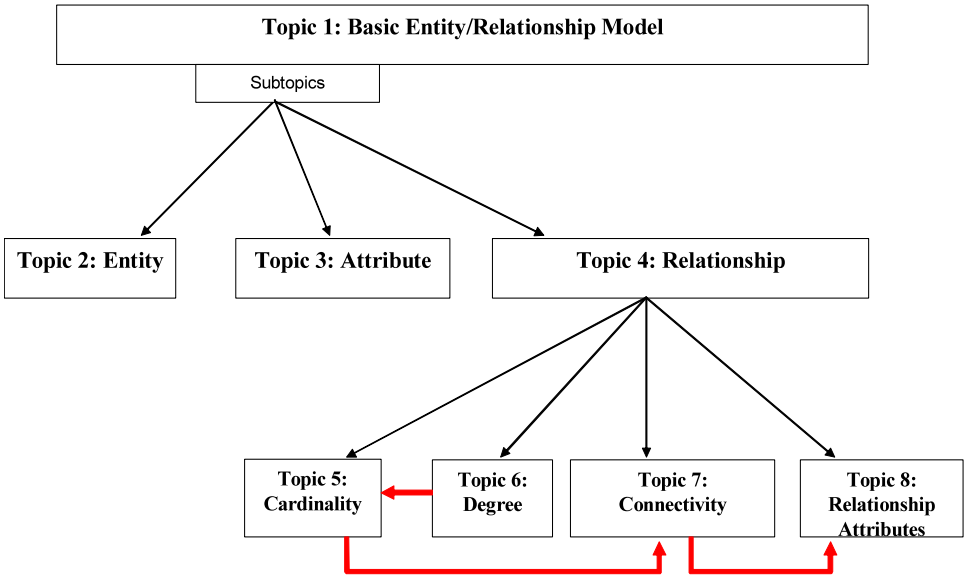


**Fig. 5** Example of Database Design AIES knowledge model

section indicated that the differences among the learning curves in each figure are statistically significant with a confidence level of 95%.

It is important to notice that, although we experiment with simulated students, real situations are evaluated. Thus, simulated students interact with the system and, at the same time, the system updates its teaching strategies in order to adapt to each student, just as interactions with real students are supposed to happen.

### *5.1.1 RLATES domain model*

For the experiments in this work, we use the knowledge model shown in Fig. 5. This example presents eight topics (knowledge items) and fifty-two tasks, where most of the topics have two *definitions* tasks, two *introductions* tasks and four *examples* tasks, all of them in two different formats: *text* and *video*. The red arrows in this figure represent *is-prerrequisite* relationships between topics. Note that this information is only used for the description of the simulated students behaviour for the experimentation of this paper (see Sect. 5.1.3), but this kind of information is not necessary for the system in order to sequence the AIES content. We propose to learn the sequence of AIES contents by using the Reinforcement Learning model (see Sect. 4), learning good pedagogical strategies for each group of students based only on previous interaction with other students with similar learning characteristics, not based on information from a human expert.

### *5.1.2 RLATES interface model*

In the experiments presented in this paper, we have not used the interface model, because the system has interacted with simulated students. However, two issues to keep in mind are introduced. Firstly, the AIES decides the sequence and the format of the material in order to find the best possible way to show the contents to the students according to the student needs. This is a pedagogical module concern, because the reinforcement learning model applied to the AIES is able to decide how the student prefers an item of knowledge to be explained (format, sequence, etc.).

Secondly, it is necessary to maintain the attention of the student at every moment, providing him/her with the sensation of control of the interaction with the system. This issue involves the interface and pedagogical modules. The system provides the student with the possibility of learning by Direct Guidance (DG), a hypermedia technique for adaptation to the user [29]. In this technique, it is the system that chooses the next topic to show to the student in a specific format when the student pushes the *next* link. In this paper, we propose to modify the traditional DG technique, providing the student with more than one Web page to visit next. When the student pushes the *next* button in RLATES, the system provides the student with several actions to execute next (several Web pages to show next). Moreover, the system provides the student with information on how much it recommends visiting each Web page next, according previous interactions with other student with similar learning characteristics using the *Boltzmann* probability distribution introduced in Sect. 2.2.3. In this way, the student is the responsible for his/her own learning, choosing the next page to visit based on the advice of the system.

### *5.1.3 Simulated students*

In this work, three clusters of students have been defined for the experimentation, depending on only two characteristics

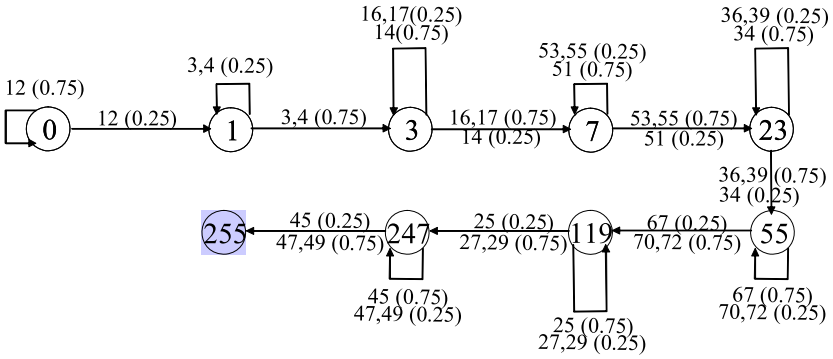


**Fig. 6** MDP for cluster 2 students learning the reduced domain module shown in Fig. 5

**Table 3** Student clusters and their characteristics

| Student cluster | Tasks | Formats |
|---|---|---|
| Cluster 1 | Definitions (100%) | Image (100%) |
| Cluster 2 | Definitions (25%) and exercises (75%) | Image (100%) |
| Cluster 3 | Definitions (25%) and introductions (25%) and exercises (50%) | Image (75%) and text (25%) |

(in order to simplify the example): what kind of tasks must be used in order for them to learn (*definition, introduction* or *exercise*) and which format the material must be shown in (*video* or *text*). Table 3 shows the three clusters of students used in the experimentation in order to simulate the student's behaviour. A 100% *cluster 1* students require the *definition* task and the *image* format to learn. *Cluster 2* students require, like *cluster 1* students, the *image* format, but 75% of them learn only with *exercise* tasks and the other 25% learn only with *definition* tasks. Finally, *cluster 3* students may require *definition*, *introduction* and *exercise* tasks; *definitions* and *introductions* will be learned with a probability of 25% each and *exercises* with a probability of 50%. With respect to the page format, 75% of the students will learn with the *image* format and 25% of them with the *text* format. Therefore, the *cluster 1* is the most homogeneous student cluster and the *cluster 3* is the most heterogeneous student cluster.

Each cluster of students with their learning characteristics can be considered for the construction of simulated students for the experimental phase of this proposal. In this paper, we have defined the behaviour of the simulated students as *Markov Decision Process (MDP)* based on the cluster definition, where it does not matter which states the student has visited, but only the current state of the student is necessary to know if s/he is able to learn the Web page's content. The MDP has been represented by circles and arcs. The MDP states are represented by circles with the number of the state within, and the actions are represented by arcs between states. The first number that appears on the arc is the number of the action that produces the state transition, whereas the number that appears in parentheses is the probability of this transition. Different actions can share the same arc. That means that the state transition can be produced for any of the actions in the arc with their probability associated.

An example of an MDP for a cluster 2 student in the domain model of Fig. 5 is shown in Fig. 6, where only the actions that produce state transitions appear. It is supposed that the student does not change his/her state when a different action (an action that does not appear in the state) is carried out. For the construction of the MDP of the clusters, the *is-prerequisite* relationships between topics shown in Fig. 5 are taken into account. For instance, one student could not learn the *Cardinality* topic if s/he does not yet know the *Degree* topic. In Fig. 6 we can see how when the student is in the *state 0* and the *action 12* is carried out, the student will stay at the same state with a probability of 75% and will change his/her state to the *state 1* with a probability of 25%.

### 5.2 Experimental results varying the system learning parameters

Firstly, in this experiment sequence, we extend the experimentation carried out in [30] varying some of the learning parameters of the system. All of these experiments show how these parameters affect the convergence of the AIES learning, dealing with the goal of the AIES: first, to converge; second, to minimize the time spent in the teaching process (measured in number of students needed in order to converge to a near optimal policy); and third, to minimize the number of actions carried out by the pedagogical policy learned.

The learning variables used are: *the learning rate parameter* ($\alpha$) of the $Q$ update equation (see (6)), which varies from 0.9 (the system learns quickly) to 0.1 (the system learns slowly); and *the homogeneity of the student cluster*, and hence, the determinism of the domain. The simulated students defined in Table 3 have interacted with the system in order to test its convergence. Cluster 1 is the most deterministic cluster, because all of the Cluster 1 students have similar learning characteristics and Cluster 3 is the most stochastic one, given that the students learn the system content

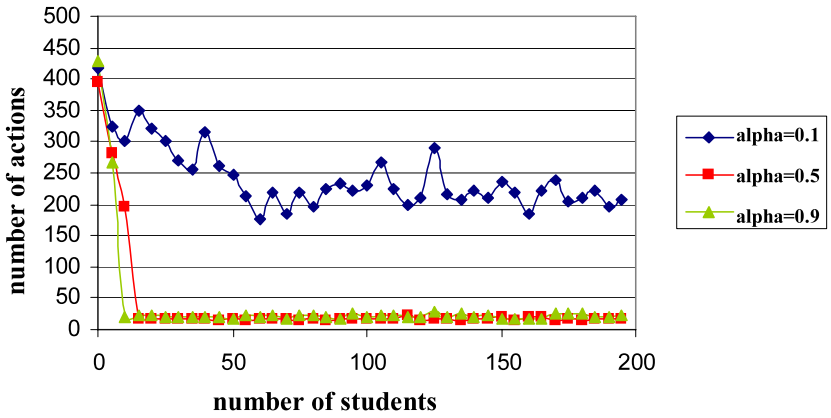


**Fig. 7** AIES learning curve for cluster 2 students and $e = 0.9$. Curve varying the learning rate parameter ($\alpha$)

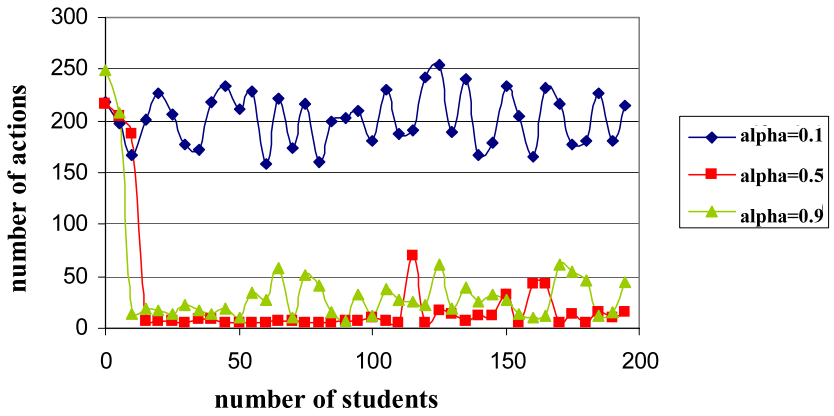


**Fig. 8** AIES standard deviation curve for cluster 2 students and $e = 0.9$, varying the learning rate parameter ($\alpha$)

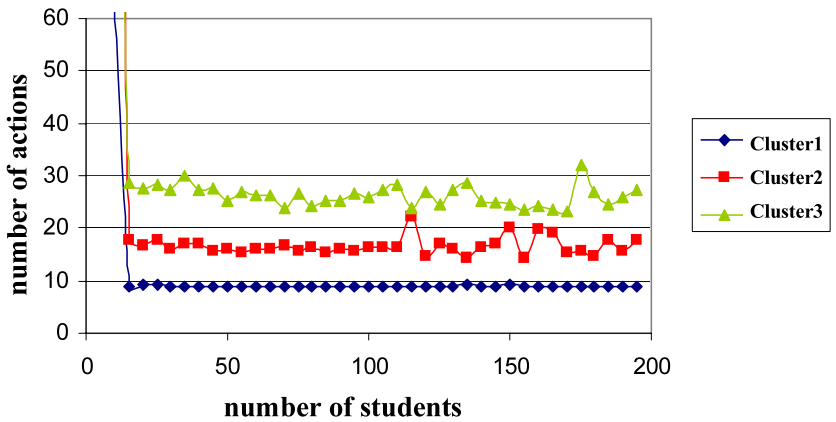


**Fig. 9** AIES learning curve when different clusters students interact with the system. $e = 0.9$ and $\alpha = 0.5$

in different ways. In these experiments, the *e-greedy* exploration/exploitation strategy has been implemented and the $e$ exploration parameter has been fixed at 0.9.

Figures 7 and 8 summarize the learning performance for different values of $\alpha$ when Cluster 2 students interact with the system. Figure 7 shows that when the value of $\alpha$ decreases, the system behaviour is worse. For instance, when $\alpha = 0.1$, almost 65 students are needed in order for the system to converge to an action policy that requires 230 actions to teach to a student. However, when $\alpha = 0.5$, the system only needs around 15 students to converge to pedagogical strategies that only need 15 actions to be carried out. If we compare the learning curve when $\alpha = 0.5$ and $\alpha = 0.9$, we can see that they are almost similar: when $\alpha = 0.9$ the system needs fewer students to converge (only 10 students), but more actions are needed to be carried out (20 actions). On the other hand, in Fig. 8 we can see at the standard deviation curve that the system behaves in a more stable manner when $\alpha = 0.5$ than when $\alpha = 0.9$, there been fewer differences between the student interactions.

Figure 9 shows that the more heterogeneous the system is, the worse the system behaves. For instance, when clus-

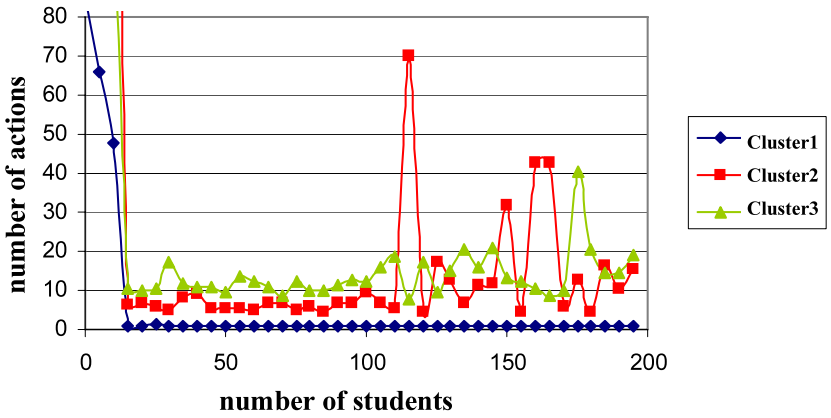


**Fig. 10** AIES standard deviation curve when different clusters students interact with the system. $e = 0.9$ and $\alpha = 0.5$

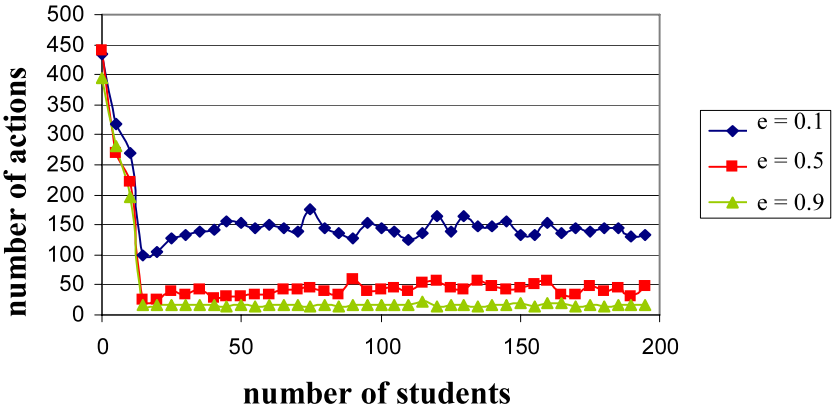


**Fig. 11** AIES learning curve for cluster 2 students and $\alpha = 0.5$. Curve varying the $e$ parameter of the *e-greedy* exploration policy

ter 1 students interact with the system, it needs only 15 students to converge carrying out an average of 8.5 actions. However, when cluster 3 students interact with the system it needs an average of 28 actions in order for the student to learn the course content. Furthermore, in Fig. 10 we can see that the more determinist the students are, the more similar their interactions are and the more stable the system behaviour is.

Figures 9 and 10 show the system performance when students of different clusters interact with the system. For the experiment, we have fixed the $\alpha$ parameter to 0.5.

After these experiments, we can conclude that the system is able to converge to almost optimal pedagogical strategies, interacting with few students. Moreover, we have demonstrated that 0.5 and 0.9 are good values for the $\alpha$ parameter. Finally, we have demonstrated that the more homogeneous the students in a cluster are, the faster and better the system converges.

### 5.3 Experimental results comparing *e*-greedy and Bolztmann exploration/exploitation policies

In this experiment sequence, the differences between the *e-greedy* and *Boltzmann* exploration/exploitation strategies are widely studied. We use the Cluster 2 students and we have fixed $\alpha$ to 0.5. Then we have modified the $e$ parameter of *e-greedy* from 0 (the system chooses the next action to carry out randomly) to 1 (the system chooses the next action trusting in the $Q$ values obtained by the previous experience). We have used values of the $\tau$ parameter of *Boltzmann* from 1 (all the actions have similar probabilities of being chosen by the system) to 0.01 (the actions have very different probabilities of being chosen based on the $Q$ table).

Figures 11 and 12 summarize the learning performance for different values of $e$ when the *e-greedy* exploration strategy has been used. In Fig. 11, the system learning convergence is shown. We can see how the higher the $e$ parameter is (more greedy is the policy used and the system exploits its knowledge more), the faster (with fewer students) and better (executing fewer actions) the system learns to behave. For instance, when $e = 0.9$ the system converges when only 15 students have interacted with it, carrying out an average of 15 actions. However, when $e = 0.1$, the system needs almost the double the amount of students (around 30 students) in order to converge; furthermore, the pedagogical strategy is not so good (it needs around 150 actions to be carried out before the students finished the interaction). Besides, we have studied the standard deviation between the interactions.

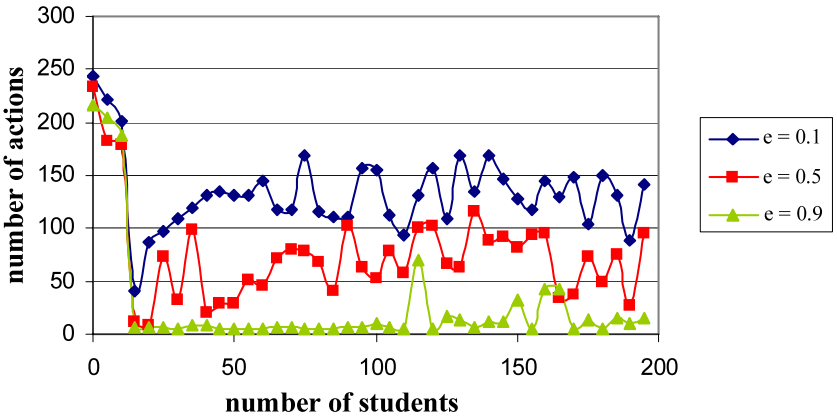


**Fig. 12** AIES standard deviation curve for cluster 2 students and $\alpha = 0.5$, varying the $e$ parameter of the *e-greedy* exploration policy

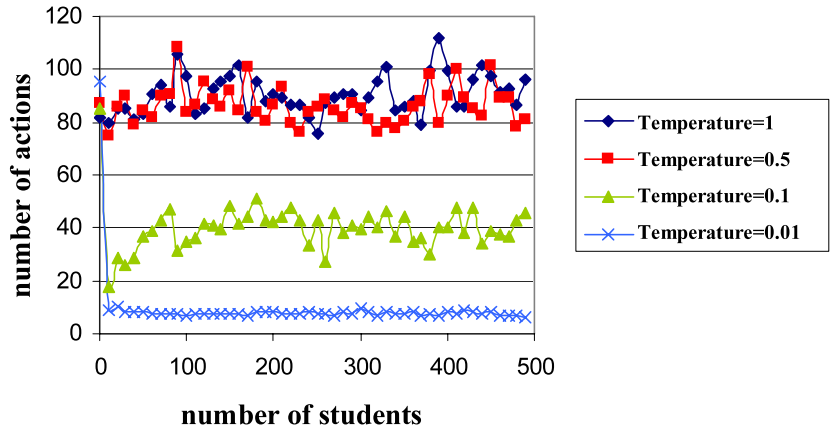


**Fig. 13** AIES learning curve for cluster 2 students and $\alpha = 0.5$. Curve varying the $\tau$ parameter of the *Boltzmann* exploration policy

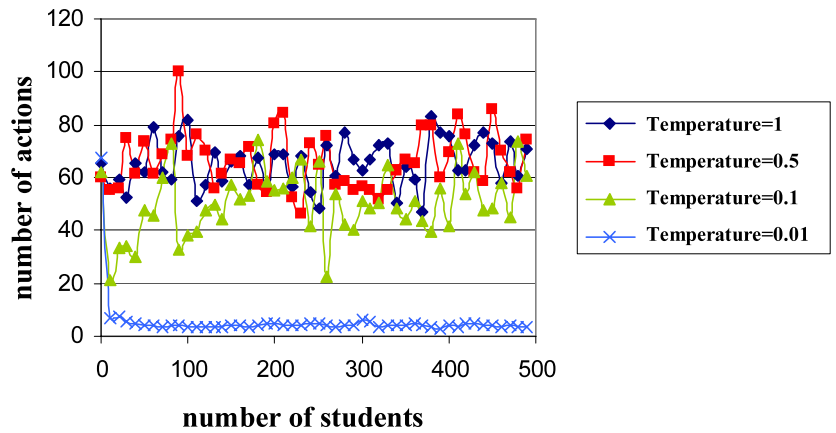


**Fig. 14** AIES standard deviation curve for cluster 2 students and $\alpha = 0.5$, varying the $\tau$ parameter of the *Boltzmann* exploration policy

In Fig. 12 we can see how the greedier the policy is, the lower the difference is between the student interactions with our system.

On the other hand, Figs. 13 and 14 show the importance of choosing a good value for the *temperature* parameter ($\tau$) of the *Boltzmann* exploration policy. In Fig. 13 we can see that the lower the value of the $\tau$ parameter, the faster and better the system learns. Moreover, in Fig. 14 the standard deviation shows that the system is more stable when the value of $\tau$ is low. For instance, when $\tau = 0.01$ the systems behavior is very good.

Then, a question is posed: if the system is able to converge with both exploitation/exploration strategies (*e-greedy* and *Boltzmann*), which of them is the most appropriate to use when real students interact with the system? In order to answer this question, first of all it is interesting to compare the system learning curves with both exploration/exploitation strategies with the values of the parameters that provide the best results. In Fig. 15 we can see how the system converges to almost 19 actions when the *e-greedy* strategy is used and it converges to 24 actions when the *Boltzmann* strategy is used. In both cases, the system con-

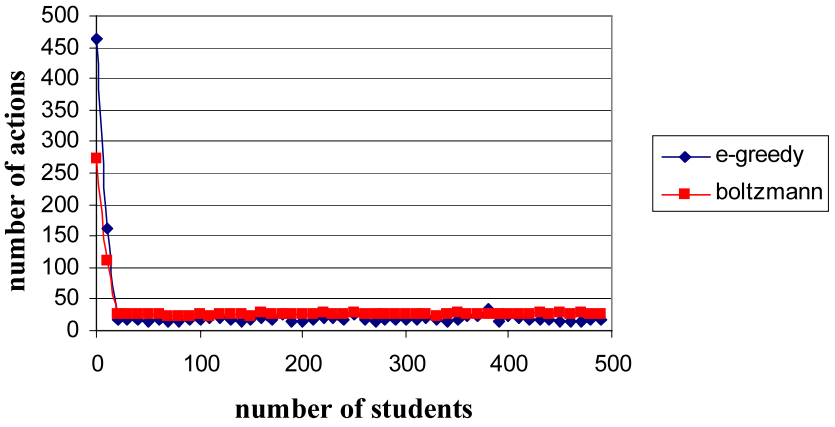


**Fig. 15** AIES learning curve comparing *e-greedy* and *Boltzmann* exploration strategies. Cluster 2 students, $\alpha = 0.5$, $\tau = 0.01$ and $e = 0.9$

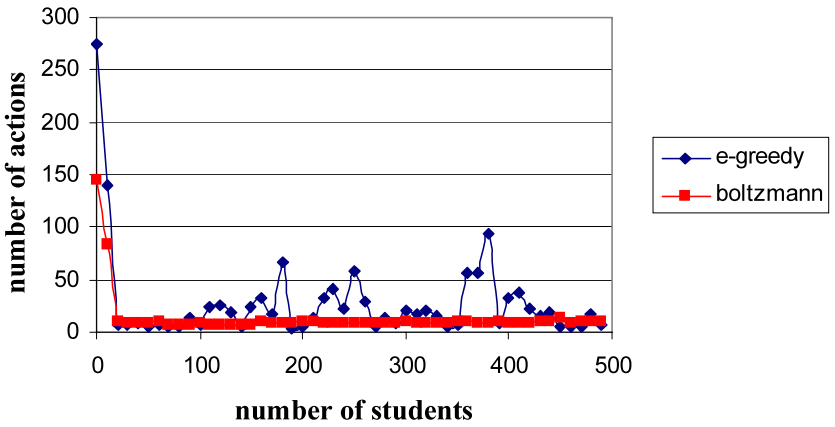


**Fig. 16** AIES standard deviation curve comparing *e-greedy* and *Boltzmann* exploration strategies. Cluster 2 students, $\alpha = 0.5$, $\tau = 0.01$ and $e = 0.9$

verges when only 20 students have interacted with it. However, in Fig. 16 we can see how the system behaves in a more stable manner when the *Boltzmann* strategy has been used, given that the standard deviation is always very low. This is a very important property on an AIES, because in this way, all the students have similar interactions with the system.

In conclusion, we can extract from these experiments that the system is able to converge with both, the *e*-greedy and the Boltzmann exploration/exploitation strategies. Moreover, we have proved that the greedier the strategy, the faster and better the system converges for this situation. Furthermore, the convergence to pedagogical strategies is nearly similar using both strategies, but there are smaller differences between the students interaction when the system uses de Boltzmann exploration strategy. This exploration strategy has been chosen for the following experiments.

### 5.4 Experimental results in attempting to reduce the system training phase

We have demonstrated that the system is able to learn almost optimal pedagogical policies training the system with few students when, initially, it has no knowledge at all of pedagogical tactics. However, the students could get bored when the system is not teaching with a good pedagogical policy, carrying out more actions than students need.

Then, a new question arises: is it possible to reduce the number of students necessary for the system training?

Reducing the training phase is desirable because during this phase the students interacting with the system do not learn in the best way because the system has not yet learned to teach.

Sometimes in educational systems, simulated students have been used in order to train teachers [31]. Moreover, in RL problems, the systems are sometimes initialized with information from previous interactions [32]. In this experimental sequence we propose to initialize the $Q$ table of the system with information coming from the interaction with other students with different learning characteristics.

In this sense, there are two possibilities:

1. If the current student has the same learning characteristics as the previous students, the system does not need to adapt its $Q$ table, so the size of the system *training* phase is equal to 0.
2. Otherwise, the system needs to adapt its $Q$ table according to the learning characteristics of the current student.

The following experiments are going to prove that by initializing the $Q$ table we can reduce the system *training* phase,

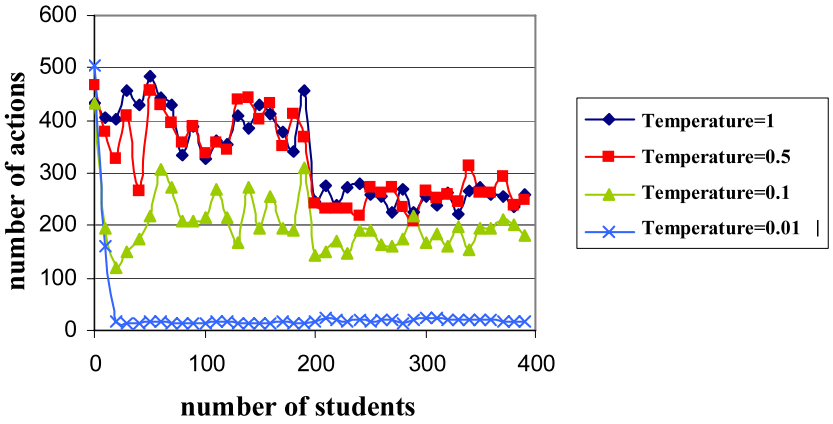


**Fig. 17** AIES learning curve initializing the system $Q$ table with cluster 3 students. Then cluster 2 students begin to interact with the system. $\alpha = 0.5$ and $\tau = 0.01$

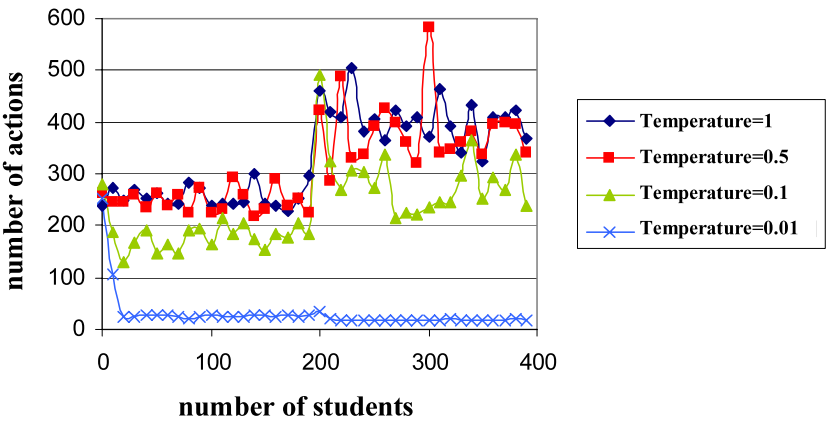


**Fig. 18** AIES learning curve initializing the system $Q$ table with cluster 2 students. Then cluster 3 students begin to interact with the system. $\alpha = 0.5$ and $\tau = 0.01$

even when this initialization is not the best for the current student cluster. In order to carry out these experiments, we have initialized the system pedagogical strategies (its $Q$ table) interacting with enough students of a specific cluster, $X$. Then, when the system has converged to a near optimal pedagogical strategy for *cluster X* students, learners with different learning characteristics, belonging to another cluster, begin to interact with the system. Then, we can analyze the size of the system *training* phase and if the system is able to learn an optimal pedagogical strategy. Similarly to experiments carried out in Sect. 5.3, in these experiments we have varied only the *temperature* of the *Boltzmann* exploitation strategy.

Results are summarized in Figs. 17 and 18. On the one hand, in Fig. 17 we can see how the system converges to good pedagogical strategies for cluster 3 students interacting with 20 students; then cluster 2 students begin to interact with the system and only 10 students are needed in order for the system to adapt its pedagogical strategy according to the learning characteristics of the new kind of students. On the other hand, in Fig. 18 we can see the reverse experiment; first the system has been initialized with cluster 2 simulated students and then cluster 3 students begin to interact with the system; the results obtained are similar to the previous experiment.

These experiments are summarized in Fig. 19, presenting the AIES learning curve when the system has been previously initialized and when the system has not been previously initialized. This figure shows the main conclusions extracted from these experiments: how the system training phase has been reduced to the half when the system is initialized. Moreover, sometimes the pedagogical strategy is better, as in cluster 2, carrying out fewer actions when the system has previously explored the appropriate actions for the current cluster of students, although sometimes it could be worse, as in cluster 3, where the system has not been sufficiently explored.

The practical application of these experiments is to initialize the system with simulated students and to use it with real students.

## 6 Conclusions and further research

In this paper different experiments have been carried out by applying the Reinforcement Learning model in a Database Design (DBD) Adaptive and Intelligent Educational System (AIES). The experiments have been performed with simulated students and have been divided into three sequences of experiments, showing how the system learns to teach by

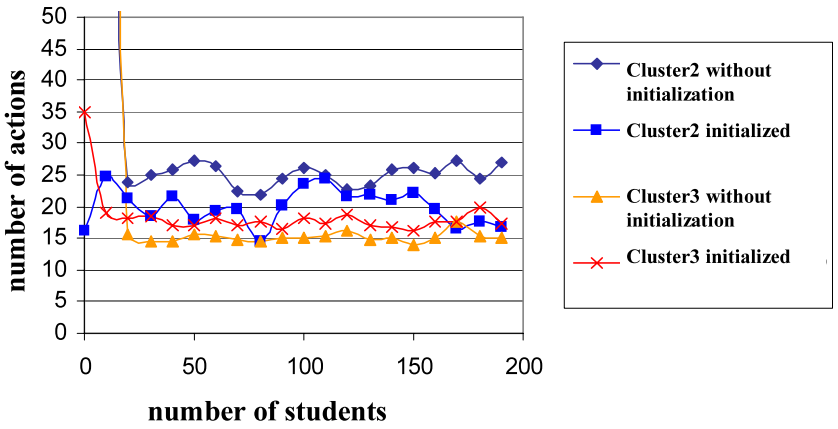


**Fig. 19** AIES learning curves summarizing the Figs. 17 and 18

trial and error at the same time as simulated students learn the AIES material. In the first sequence, the system viability has been tested by demonstrating three important issues. First, that the system converges. Second, that it converges to an almost optimal pedagogical policy, measured in number of actions that the system must carry out in order to teach all the AIES material to the student. And third, that the system does not need many students to learn to teach optimally.

In the second sequence, we have proven that choosing a good exploration and exploitation strategy is determinant for the efficacy and efficiency of the convergence of the system. Two typical exploration/exploitation strategies in RL problems have been studied in order to analyze the differences between them when the system teaches the material of the course to simulated students: the *e-greedy* exploration policy and the *Boltzmann* exploration strategy. After collecting the results of these experiments, we can conclude that using both strategies provides accurate pedagogical policies, but that the *Boltzmann* exploration/exploitation strategy achieves more stable learning curves.

Finally, in the third sequence, we have proven that initializing the AIES pedagogical strategies reduces the system *training* phase, even when this initialization is not the best for the current cluster of students.

Currently, we are involved in empirically prove real situations where, if we assume that a group of students can be modelled as a Markov Decision Process (MDP), or at least, that a set (or cluster) of students behaves following an unknown and underlying MDP then, could the pedagogical module of an AIES be developed automatically through Reinforcement Learning?

## References


1. Brusilovsky P (1999) Adaptive and intelligent technologies for Web-based education. Kunstl Intell 4:19–25. Special Issue on Intelligent Tutoring Systems and Teleteaching
2. Lebowitz M (1987) Experiments with incremental concept formation: Unimen. Mach Learn 2:103–138
3. Murray RC, VanLehn K, Mostow J (2001) A decision-theoretic approach for selecting tutorial discourse actions. In: Proc of the NAACL workshop on adaptation in dialogue systems, pp 41–48
4. Iglesias A, Martínez P, Aler R, Fernando F (2002) Applying reinforcement learning in intelligent tutoring systems. In: Proc of international conference on new educational environments (ICNEE), Lugano (Switzerland), pp 11–14
5. Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
6. Wenger E (1987) Artificial intelligence and tutoring systems. Morgan Kaufmann, San Mateo
7. Murray T (1999) Authoring intelligent tutoring systems: an analysis of the state of the art. J Artif Intell Educ 10:98–129
8. Andre E, Finkler W, Graf W, Schauder A, Wahister W (1993) Intelligent multimedia presentations. In: Maybury MT (ed) The automatic synthesis of multimodal presentations. MIT Press, Cambridge
9. Khan T, Yip YJ (1996) Pedagogical principles of case-based cal. J Comput Assist Learn 1(12):172–192
10. Sleeman A (1977) A system which allows student to explore algorithms. In: Proceedings of international joint conference on artificial intelligence, pp 780–786
11. Woolf BP (1987) Representing complex knowledge in an intelligent machine tutor. Comput Intell 3:45–55
12. Beck J (2001) ADVISOR: A machine learning architecture for intelligent tutor construction. PhD thesis, University of Massachusetts Amherst
13. Burns H, Capps C (1998) Foundations of intelligent tutoring systems: an introduction. Foundations of intelligent tutoring systems. Lawrence Erlbaum Associates, Hillsdale, pp 1–19
14. Linton F, Schaefer HP (2000) Recommender systems for learning: building user and expert models through long-term observation of application use. User Model User-Adapt Interact 10:181–208
15. Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: a survey. Int J Artif Intell Res 4:237–285
16. Puterman ML (1994) Markov decision processes: discrete stochastic dynamic programming. Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley, New York
17. Watkins CJCH (1989) Learning from delayed rewards. Phd Thesis, King's College, Cambridge
18. Thrun SB (1992) The role of exploration in learning control. In: White DA, Sofge DA (eds) Handbook of intelligent control: neural, fuzzy and adaptive approaches. Van Nostrand Reinhold, New York
19. Elmasri R, Navathe SB (1994) Fundamentals of database systems, 2nd edn. Benjamin/Cummings, Redwood City
20. Rich E (1979) User modelling via stereotypes. Cogn Sci 4:329–354

21. Carr B, Goldstein I (1977) Overlays: A theory of modelling for computer aided instruction. Technical Report ai memo 406, AI Laboratory, Massachusetts Institute of Technology, Cambridge
22. VanLehn K, Zhendong N (2001) Bayesian student modelling, user interfaces and feedback: a sensitivity analysis. Int J Artif Intell Educ 2:155–184
23. Iglesias A, Martínez P, Aler R, Fernando F (2002) Learning to teach database design by trial and error. In: Proc of 4th international conference on enterprise information systems (ICEIS), Ciudad Real (Spain), pp 500–505
24. Sison R, Shimura M (1998) Student modeling and machine learning. Int J Artif Intell Educ 9:128–158
25. Quinlan JR (1993) C4.5 Programs for machine learning. Morgan Kaufmann, San Mateo
26. Ur S, VanLehn K (1995) STEPS: A simulated, tutorable physics student. J Artif Intell Educ 6(4):405–437
27. VanLehn K, Ohlsson S, Nason R (1994) Applications of simulated students: An exploration. J Artif Intell Educ 5(2):135–175
28. Montgomery DC (2005) Design and analysis of experiments, 5th edn. Wiley, New York
29. Brusilovsky P (1996) Methods and techniques of adaptive hipermedia. User Model User Adapt Interact 6(2–3):87–129
30. Iglesias A, Martínez P, Aler R, Fernando F (2003) An experience applying reinforcement learning in a web-based adaptive and intelligent education system. Inform Educ Int J 2:1–18
31. Doak ED, Keith M (1986) Simulation in teacher education: The knowledge base and the process. Tenn Educ 16(2):14–17
32. Carroll JM, Peterson ST (2002) Fixed vs dynamic sub-transfer in reinforcement learning. In: Proc ICMLA 2002, M Arif Wani, Las Vegas, Nevada
33. Stankov S, Božičević J (1998) The computer tutor in the new model of learning and teching control principles. In: Proc of 1st international workshop: mechatronics and industrial engineering, Bratislava, Slovak Republic, pp 78–88
34. VanLehn K, Zhendong N (2001) Bayesian student modelling, user interfaces and feedback: a sensitivity analysis. Int J Artif Intell Educ 2:155–184
35. Aleven V, Koedinger KR (2000) The need for tutorial dialog to support self-explanation. In: Building dialogue systems for tutorial applications, papers of the 2000 AAAI Fall Symposium, pp 65–73
36. Kumar R, Rosé CP, Aleven V, Iglesias A, Robinson A (2006) Evaluating the effectiveness of tutorial dialogue instruction in an exploratory learning context. Intell Tutor Syst 2066:666–674
37. Brusilovsky P, Schwarz E, Weber G (1996) Elm-art: An intelligent tutoring system on world wide web. In: Frasson C, Gauthier G, Lesgold A (eds) Intelligent tutoring systems, vol 1086. Springer, Berlin, pp 261–269
38. André E, Müller J, Rist T (1996) WIP/PPP: Automatic generation of personalized multimedia presentations. In: Proc of Multimedia 96, 4th ACM International Multimedia Conference, Boston, pp 407–408
39. Anderson J, Reiser B (1985) The lisp tutor. Byte 10(4):159–175
40. Prentzas J, Hatzilygeroudis I, Garofalakis J (2002) A web-based intelligent tutoring system using hybrid rules as its representational basis. In: Cerri SA, Gouarderes G, Paraguacų F (eds) Proceedings of the 6th international conference, ITS 2002. Lecture notes in computer science, vol 1. Springer, Berlin, pp 119–128
41. Rich E (1979) User modelling via stereotypes. Cogn Sci 3(4):329–354
42. Carr B, Goldstein I (1977) Overlays: A theory of modelling for computer aided instruction. Technical Report ai memo 406, AI Laboratory, Massachusetts Institute of Technology, Cambridge
43. Langley P, Ohlsson S (1984) Automated cognitive modelling. In: Proceedings of the second national conference on artificial intelligence
44. Baffes P, Mooney R (1996) Refinement-based student modelling and automated bug library construction. J Artif Intell Educ 7(1):75–116
45. Sison R, Nũmao M, Shimura M (2000) Multistrategy discovery and detection of novice programmer errors. Mach Learn 38:157–180
46. Sleeman D, Brown S (1982) Intelligent tutoring systems. Computers and people series. Academic Press, London
47. Hoppe U (1994) Deductive error diagnosis and inductive error generalization for intelligent tutoring systems. J Artif Intell Educ 5:27–49
48. Beck J (2001) ADVISOR: A machine learning architecture for intelligent tutor construction. PhD thesis, University of Massachusetts Amherst

**Ana Iglesias Maqueda** is a faculty of the Computer Science Department of Universidad Carlos III de Madrid, since February 2006. She got a degree in Computer Science from Universidad Carlos III of Madrid in 1999 and she received the Ph.D. degree in Computer Science from University Carlos III of Madrid (UC3M) in 2004. Since January 2005 until August December she was in a postdoctoral stay at the Human Computer Interaction Institute of Carnegie Mellon University. Since 2000, she works at the Advanced Databases Group in the Computer Science Department at Universidad Carlos III of Madrid. She is currently teaching Database Design and Advanced Databases. She has been working in several National research projects about Advanced Database Technologies, CASE environments, Natural Language Processing and Information Retrieval. Her research interests include Adaptive Intelligent Educational Systems, Machine Learning, Natural Language Processing, Information Retrieval and Database Design.

**Paloma Martínez Fernández** got a degree in Computer Science from Universidad Politécnica of Madrid in 1992. Since 1992, she is the leader of the Advanced Databases Group in the Computer Science Department at Universidad Carlos III of Madrid. In 1998 she obtained the Ph.D. degree in Computer Science from Universidad Politécnica of Madrid. She is currently teaching Database Design and Advanced Databases in the Computer Science Department at the Universidad Carlos III de Madrid. She has been working in several European and National research projects about Natural Language Processing, Information Retrieval, Question Answering and Advanced Database Technologies.

**Ricardo Aler Mur** is associate professor at the Computer Science Department at Universidad Carlos III de Madrid. He received his PhD degree in Computer Science from Universidad Politécnica de Madrid in 1999 and his MSc in Decision Support Systems from Sunderland University (UK) (1993). He has worked in several research national and European projects related to automated planning, process optimization, and evolutionary computation. His current research interests are focused on Machine Learning, Genetic Programming, Evolutionary Computation and Brain-Computer Interfaces.

**Fernando Fernández Rebollo** is a faculty of the Computer Science Department of Universidad Carlos III de Madrid, since October 2005. He received his Ph.D. degree in Computer Science from University Carlos III of Madrid (UC3M) in 2003. He received his B.Sc. in 1999 from UC3M, also in Computer Science. From 2001, he became assistant and associate professor at UC3M. In the fall of 2000, Fernando was a visiting student at the Center for Engineering Science Advanced Research at Oak Ridge National Laboratory (Tennessee). He was also a postdoctoral fellow at the Computer Science Department of Carnegie Mellon University since October 2004 until December 2005.

He is the recipient of a pre-doctoral FPU fellowship award from Spanish Ministry of Education (MEC), a Doctoral Prize from UC3M, and a MEC-Fulbright postdoctoral Fellowship. He has more than 30 journal and conference papers, mainly in the field of machine learning and planning.

He is interested in intelligent systems that operate in continuous and stochastic domains. In his thesis, entitled “Reinforcement Learning in Continuous State Spaces”, he studied different discretization methods of the state space in Reinforcement Learning problems, specifically Nearest Prototype approaches. When he arrived to CMU, he focused his research on the transfer of policies between different Reinforcement Learning tasks, and in how to bias the exploration of new learning processes with previously learned policies. Currently, his research is also focused on connecting classical planning methods with learning mechanism that allow to behave in dynamic and stochastic environments. Applications of his research include robot soccer, adaptive educational systems, and tourism support tools.